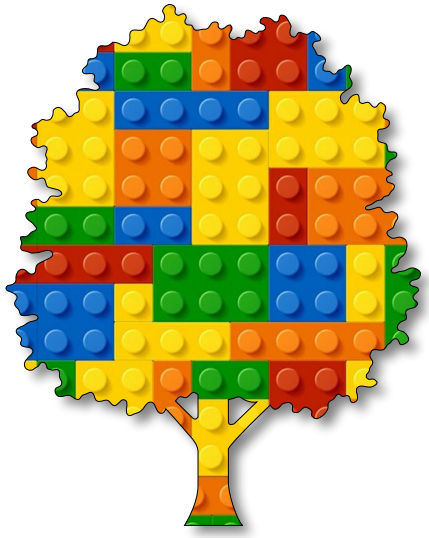
Algorithmic Art

Module D
Unit #3
array of objects



Content

Module D Unit #3 array of objects

Introduction to array of objects

In this unit, we will mash together arrays and classes with a dash of mouse functions.

Recap: Arrays are extremely useful and powerful in so many ways. An array is a list of elements inside square brackets separated by a comma. The numbering of each element starts at zero; they are called the index number. Let's create an array called num and put some numbers in it.

```
let num = [42, 56, 63, 79, 88]
```

There are five numbers (elements) in the array. They are, in order: the first one is 42, the second one is 56, the third one is 63, the fourth one is 79, and the fifth one is 88.

However, to reference the first element (42), we say it is at index 0. The second element (56) is at index 1, the third (63) is index 2, the fourth (79) is index 3, and the fifth (88) is at index 4. This may seem a bit silly, but it is the way it is, so you will need to get used to it.

Sketch D3.1 bubbles

❗ create a new sketch.
We are going to make an array of objects. We start with an empty array called `bubbles`.

```
let bubbles = []


function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
}
```

Sketch D3.2 classy bubbles

We are going to put `bubble` objects in that array, but first we need to create a `Bubble` class to create some `bubbles`.

```
let bubbles = []

function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
}
```

```
class Bubble
{
  constructor()
  {

  }

  move()
  {

  }

  show()
  {

  }
}
```

Notes

Nothing to see here.

Sketch D3.3 time for an argument

We need three arguments for the bubble: the x, y (coordinates), and r (radius). The radius will be randomly generated. The x and y positions will be created with a click or drag of the mouse.

```
let bubbles = []

function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
}

class Bubble
{
  constructor(x, y, r)
  {
    this.x = x
    this.y = y
    this.r = r
  }

  move()
  {

  }

  show()
  {
    circle(this.x, this.y, this.r)
  }
}
```

🗒️ Notes

We are not moving anything because we haven't drawn a bubble. So there is nothing to see just yet; all will be revealed soon.

Sketch D3.4 drawing the bubbles

Let us draw one bubble to start with.

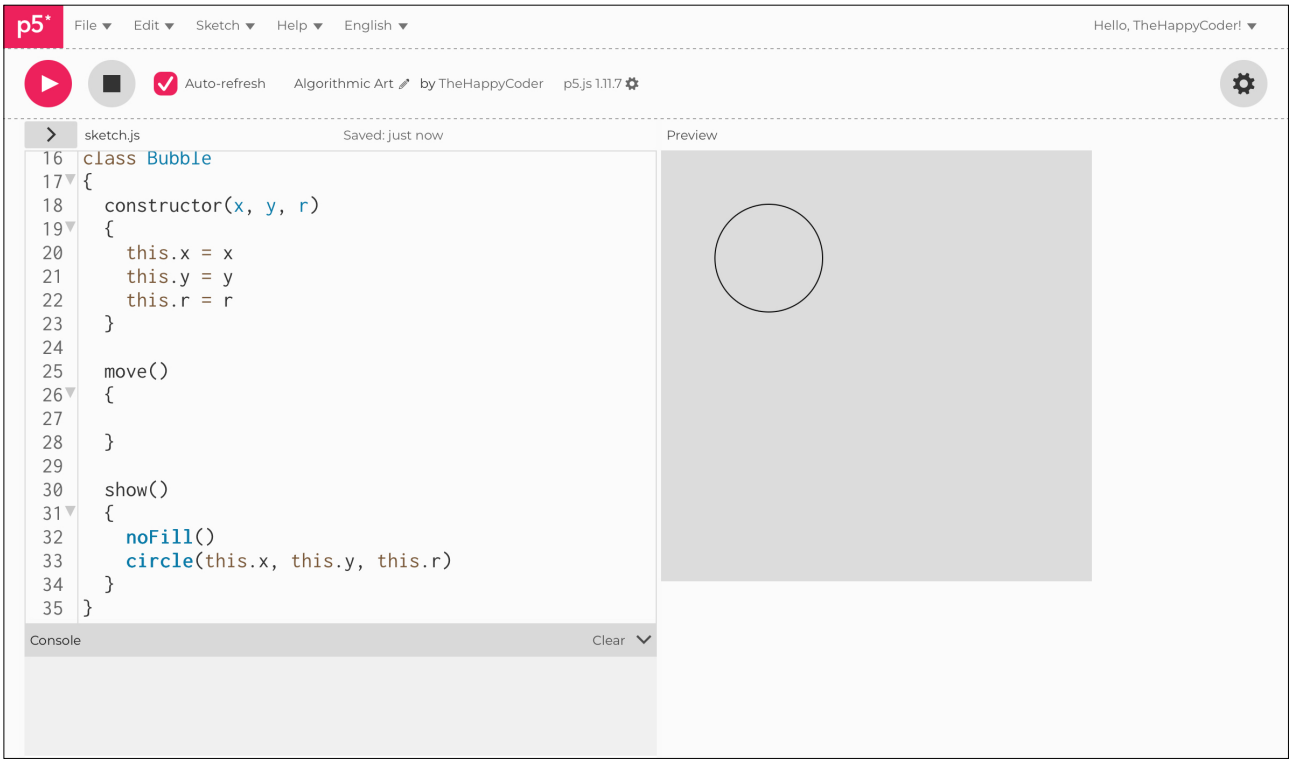
```
let bubbles = []
let bubble

function setup()
{
  createCanvas(400, 400)
  bubble = new Bubble(100, 100, 100)
}

function draw()
{
  background(220)
  bubble.show()
}

class Bubble
{
  constructor(x, y, r)
  {
    this.x = x
    this.y = y
    this.r = r
  }

  move()
  {

  }

  show()
  {
    noFill()
    circle(this.x, this.y, this.r)
  }
```

```
}
```

🗒️ Notes

We finally have something to see, our first bubble.

Figure D3.4



Sketch D3.5 random motion

Let's move it around in a random manner.

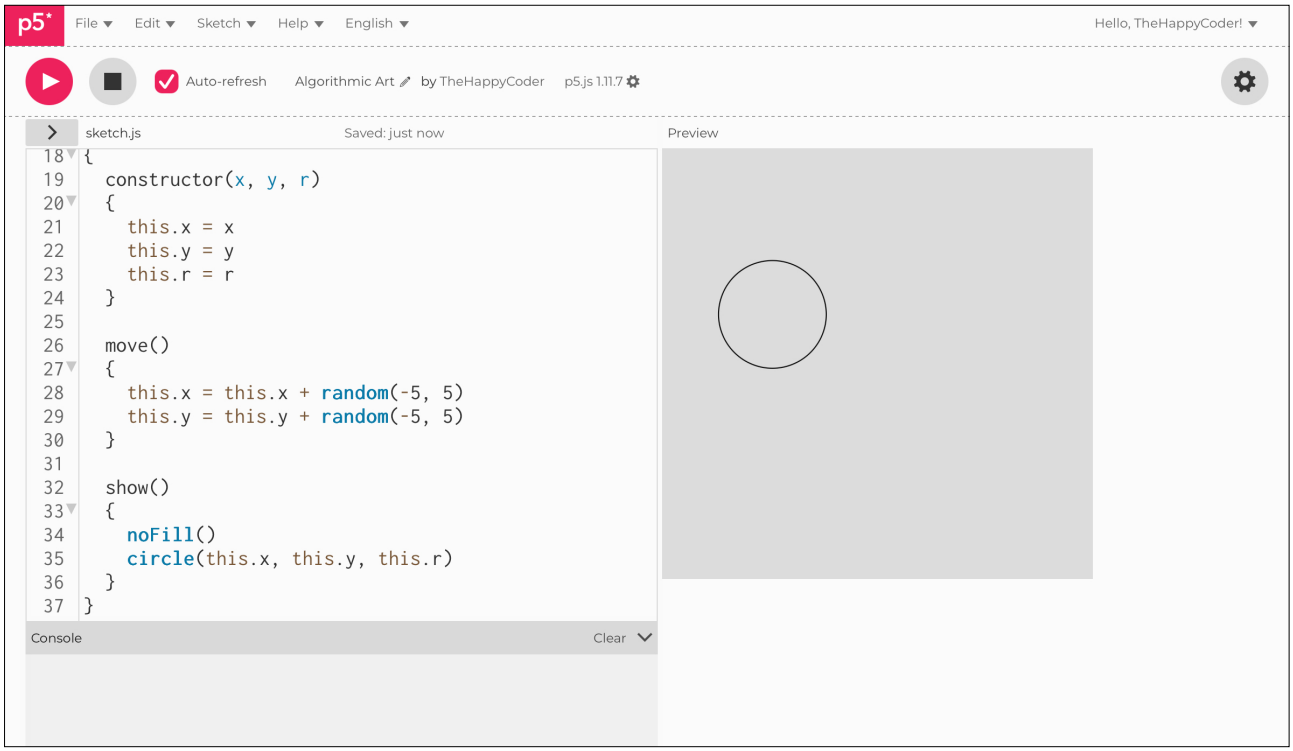
```
let bubbles = []
let bubble

function setup()
{
  createCanvas(400, 400)
  bubble = new Bubble(100, 100, 100)
}

function draw()
{
  background(220)
  bubble.show()
  bubble.move()
}

class Bubble
{
  constructor(x, y, r)
  {
    this.x = x
    this.y = y
    this.r = r
  }

  move()
  {
    this.x = this.x + random(-5, 5)
    this.y = this.y + random(-5, 5)
  }

  show()
  {
    noFill()
```

```
    circle(this.x, this.y, this.r)
  }
}
```

🗒️ Notes

What you should see is a circle wobbling around the canvas (or wandering off it eventually).

Figure D3.5



Sketch D3.6 many bubbles

We can create many of them with a nice `for()` loop and put them into the `bubbles` array. Then we have another `for()` loop to pull them out. Here we are just creating and drawing one bubble for the moment.

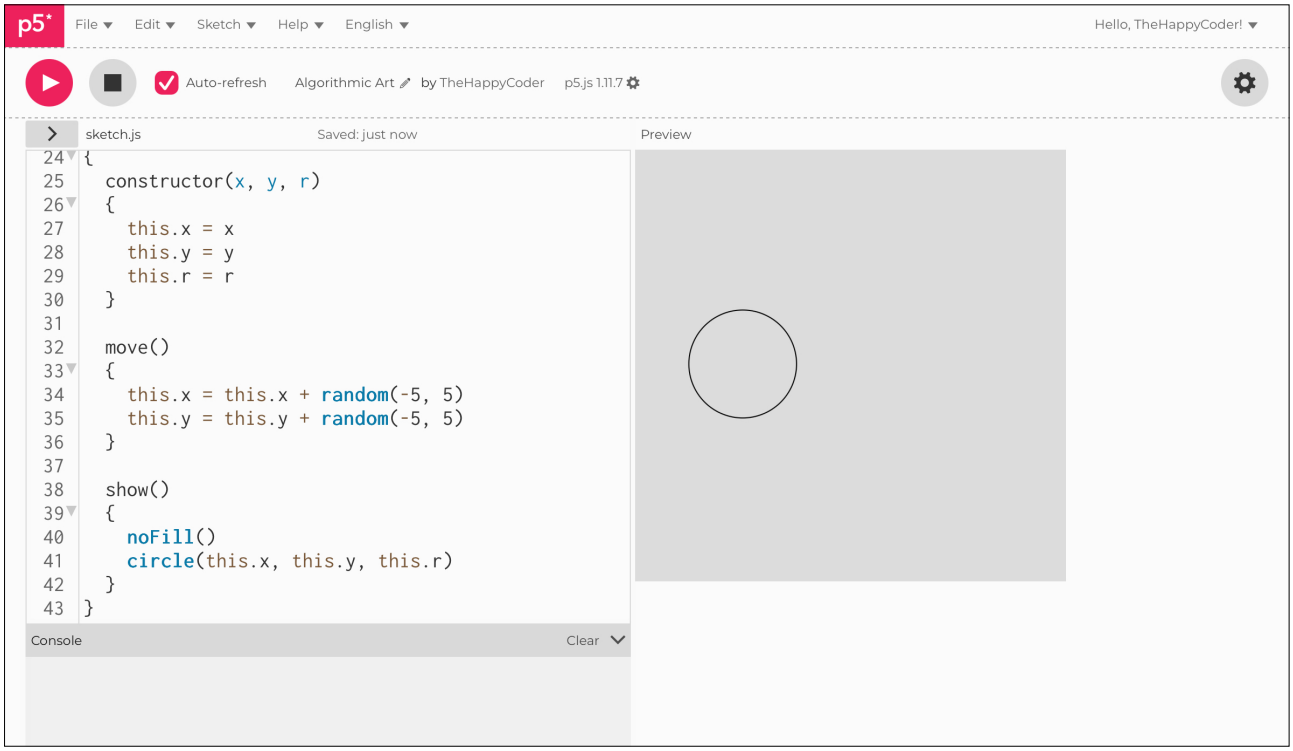
```
let bubbles = []
let bubble

function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < 1; i++)
  {
    bubbles[i] = new Bubble(100, 100, 100)
  }
}

function draw()
{
  background(220)
  for (let i = 0; i < 1; i++)
  {
    bubbles[i].show()
    bubbles[i].move()
  }
}

class Bubble
{
  constructor(x, y, r)
  {
    this.x = x
    this.y = y
    this.r = r
  }

  move()
```

```
  {
    this.x = this.x + random(-5, 5)
    this.y = this.y + random(-5, 5)
  }

  show()
  {
    noFill()
    circle(this.x, this.y, this.r)
  }
}
```

📝 Notes

We have not really changed anything; you should have the same effect.

Figure D3.6



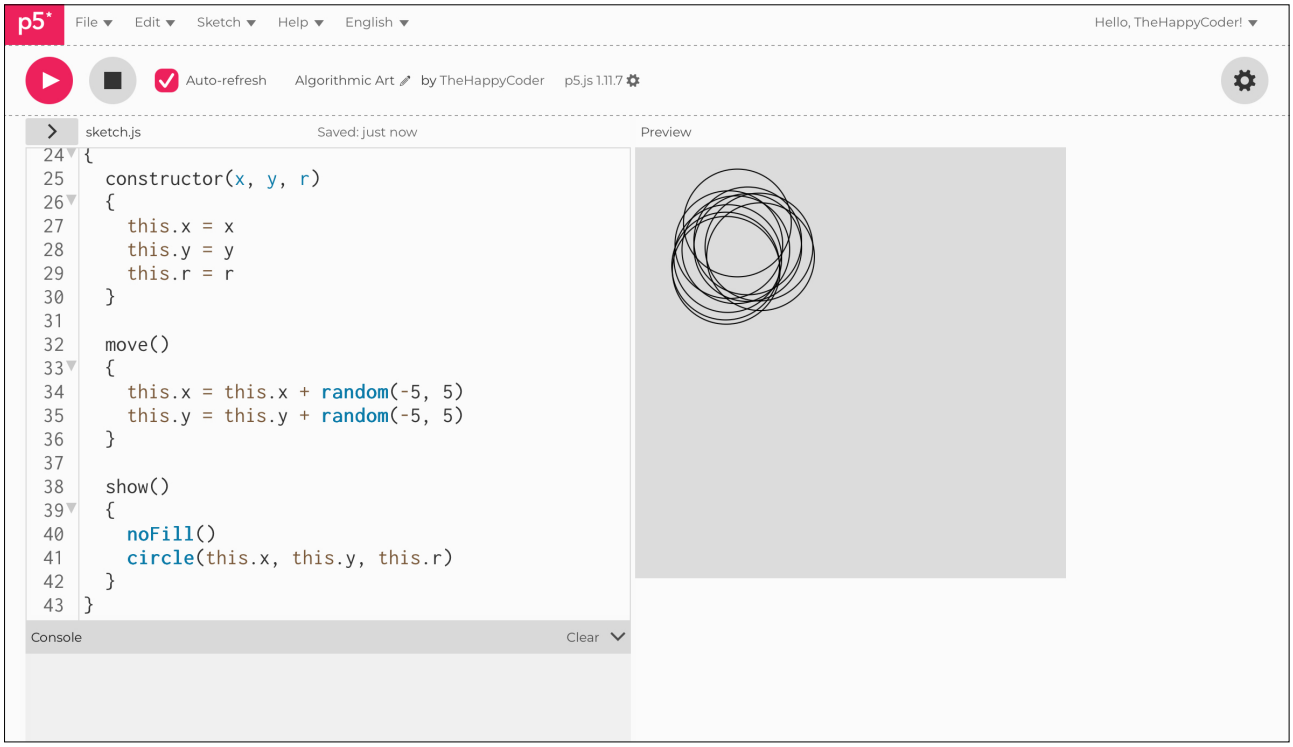
Sketch D3.7 more bubbles

We only drew one `bubble` above. There is only one `bubble` in the array of `bubbles`. We can change that by increasing it from `1` to `10`. In the second `for()` loop, we can call the length of the array rather than hardcode it just in case we change the length of the array.

```
let bubbles = []
let bubble

function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < 10; i++)
  {
    bubbles[i] = new Bubble(100, 100, 100)
  }
}

function draw()
{
  background(220)
  for (let i = 0; i < bubbles.length; i++)
  {
    bubbles[i].show()
    bubbles[i].move()
  }
}

class Bubble
{
  constructor(x, y, r)
  {
    this.x = x
    this.y = y
    this.r = r
  }

  move()
```

```
  {
    this.x = this.x + random(-5, 5)
    this.y = this.y + random(-5, 5)
  }

  show()
  {
    noFill()
    circle(this.x, this.y, this.r)
  }
}
```

Figure D3.7



Sketch D3.8 spacing them out

We started each one in the same spot. Let's space them along a line using the loop. We start at 10 and multiply the index i by 40.

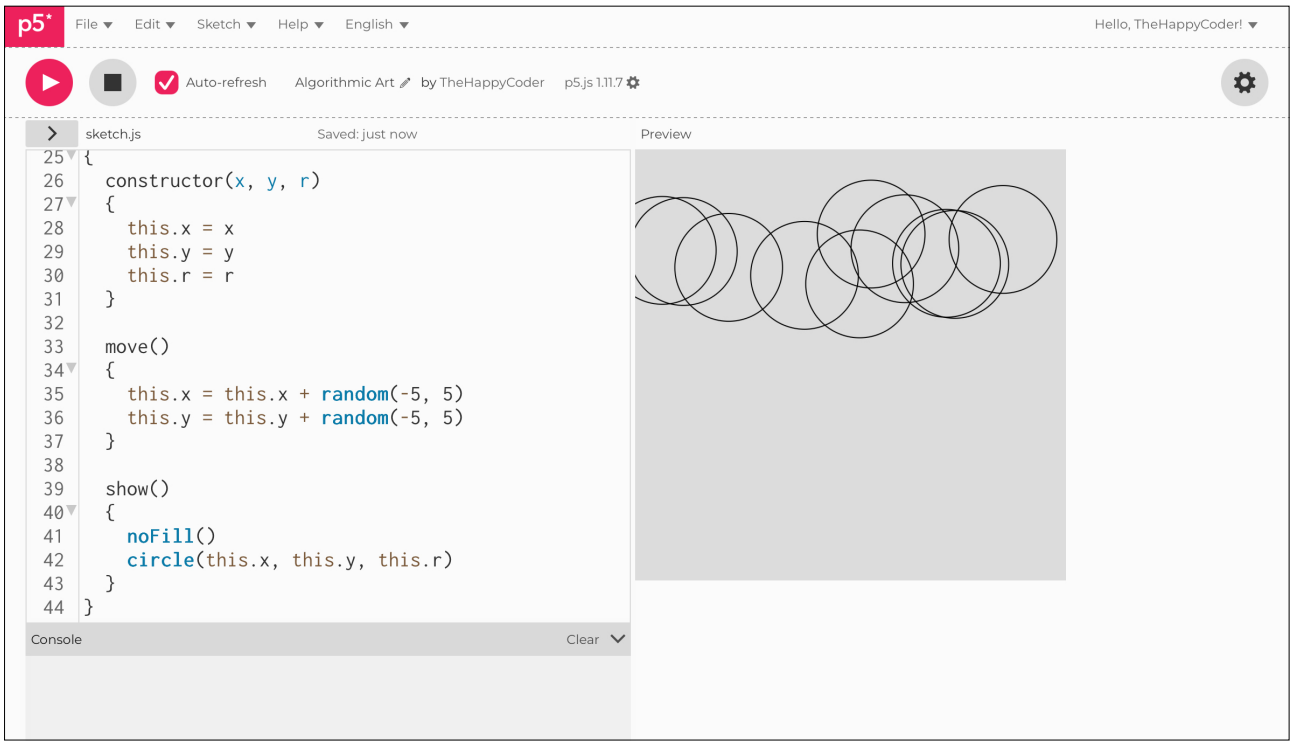
```
let bubbles = []
let bubble

function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < 10; i++)
  {
    let x = 10 + 40 * i
    bubbles[i] = new Bubble(x, 100, 100)
  }
}

function draw()
{
  background(220)
  for (let i = 0; i < bubbles.length; i++)
  {
    bubbles[i].show()
    bubbles[i].move()
  }
}

class Bubble
{
  constructor(x, y, r)
  {
    this.x = x
    this.y = y
    this.r = r
  }

  move()
```

```
  {
    this.x = this.x + random(-5, 5)
    this.y = this.y + random(-5, 5)
  }

  show()
  {
    noFill()
    circle(this.x, this.y, this.r)
  }
}
```

📝 Notes

Notice we haven't touched anything in the class itself.

Figure D3.8



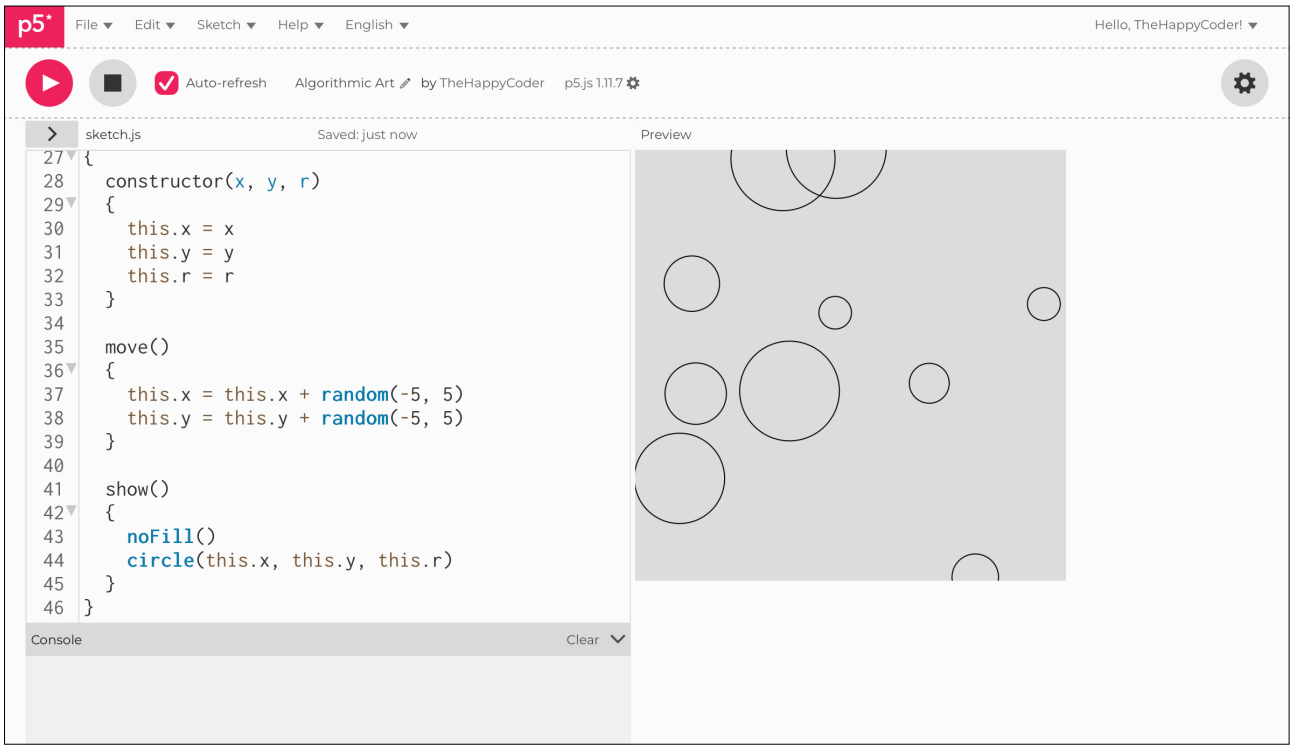
Sketch D3.9 random

Let us create a set of bubbles of random positions and sizes.

```
let bubbles = []
let bubble

function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < 10; i++)
  {
    let x = random(width)
    let y = random(height)
    let r = random(20, 100)
    bubbles[i] = new Bubble(x, y, r)
  }
}

function draw()
{
  background(220)
  for (let i = 0; i < bubbles.length; i++)
  {
    bubbles[i].show()
    bubbles[i].move()
  }
}

class Bubble
{
  constructor(x, y, r)
  {
    this.x = x
    this.y = y
    this.r = r
  }
```

```
  move()
  {
    this.x = this.x + random(-5, 5)
    this.y = this.y + random(-5, 5)
  }

  show()
  {
    noFill()
    circle(this.x, this.y, this.r)
  }
}
```

Figure D3.9



Sketch D3.10 the shimmering fog

That is all well and good, but what can we do to make it more interesting? With a few tweaks, we can do something quite slightly unexpected.

```
let bubbles = []
let bubble

function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < 1000; i++)
  {
    let x = random(width)
    let y = random(height)
    let r = random(20, 100)
    bubbles[i] = new Bubble(x, y, r)
  }
}

function draw()
{
  background(220)
  for (let i = 0; i < bubbles.length; i++)
  {
    bubbles[i].show()
    bubbles[i].move()
  }
}

class Bubble
{
  constructor(x, y, r)
  {
    this.x = x
    this.y = y
    this.r = r
  }
```

```
  move()
  {
    this.x = this.x + random(-3, 3)
    this.y = this.y + random(-3, 3)
  }

  show()
  {
    noStroke()
    fill(51, 10)
    circle(this.x, this.y, this.r)
  }
}
```

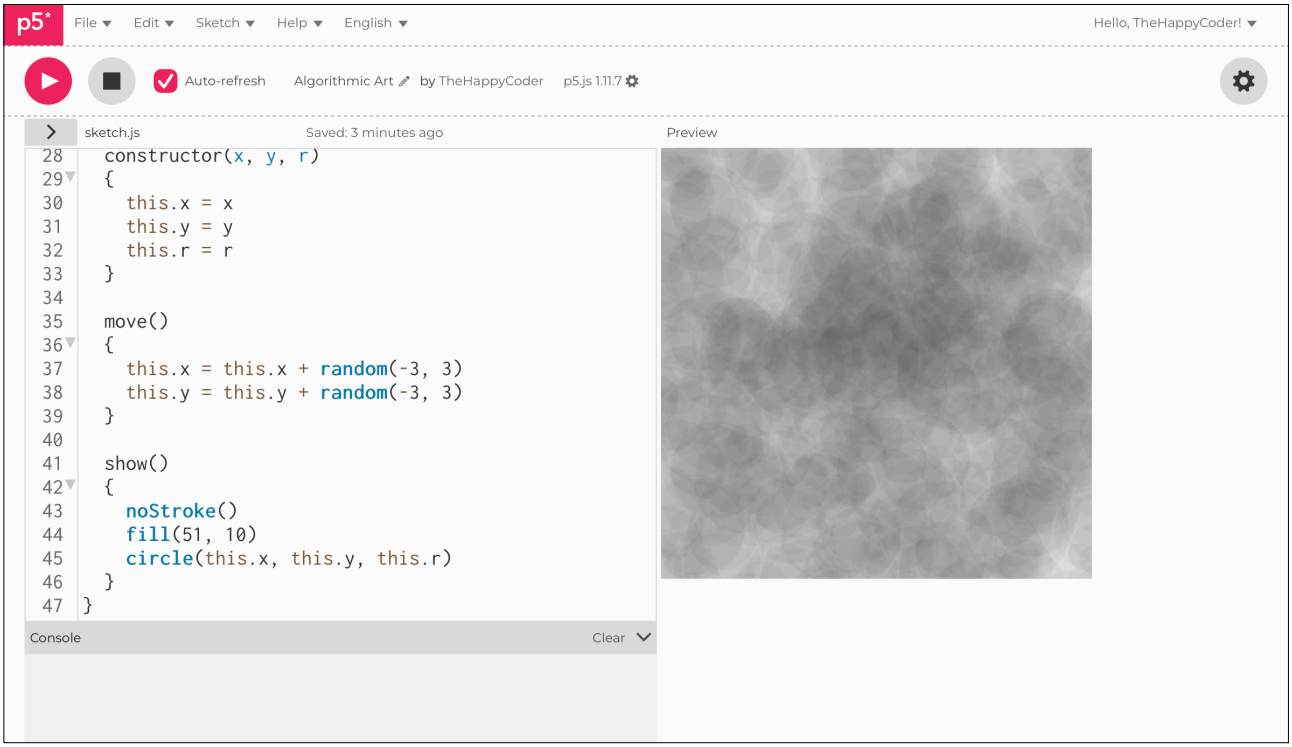
🗒️ Notes

Sometimes it doesn't take much.

🌻 Challenge

Play with it.

Figure D3.10



Sketch D3.11 simpler times

Let's go back to a simpler sketch. A bit of deleting and a couple of lines of code will bring you back to a familiar starting point.

❗ Remove the lines of code highlighted in blue and commented out (//), and add the `noFill()` in the `show()` function.

```
let bubbles = []
let bubble

function setup()
{
  createCanvas(400, 400)
  // for (let i = 0; i < 1000; i++)
  // {
  //   let x = random(width)
  //   let y = random(height)
  //   let r = random(20, 100)
  //   bubbles = new Bubble(x, y, r)
  // }
}

function draw()
{
  background(220)
  for (let i = 0; i < bubbles.length; i++)
  {
    bubbles[i].show()
    bubbles[i].move()
  }
}

class Bubble
{
  constructor(x, y, r)
  {
    this.x = x
    this.y = y
    this.r = r
```

```
  }

  move()
  {
    this.x = this.x + random(-3, 3)
    this.y = this.y + random(-3, 3)
  }

  show()
  {
    // noStroke()
    // fill(51, 10)
    noFill()
    circle(this.x, this.y, this.r)
  }
}
```

🗒️ Notes

A blank grey canvas.

Sketch D3.12 this is what you get

You should have this...

```
let bubbles = []
let bubble

function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
  for (let i = 0; i < bubbles.length; i++)
  {
    bubbles[i].show()
    bubbles[i].move()
  }
}

class Bubble
{
  constructor(x, y, r)
  {
    this.x = x
    this.y = y
    this.r = r
  }

  move()
  {
    this.x = this.x + random(-3, 3)
    this.y = this.y + random(-3, 3)
  }

  show()
```

```
  {
    noFill()
    circle(this.x, this.y, this.r)
  }
}
```

🗒️ Notes

Nothing should happen! No bubbles, we want to draw bubbles where we click the mouse.

Sketch D3.13 mouse click

Creating a new bubble at the click of a mouse, we create a function called `mousePressed()` to do this.

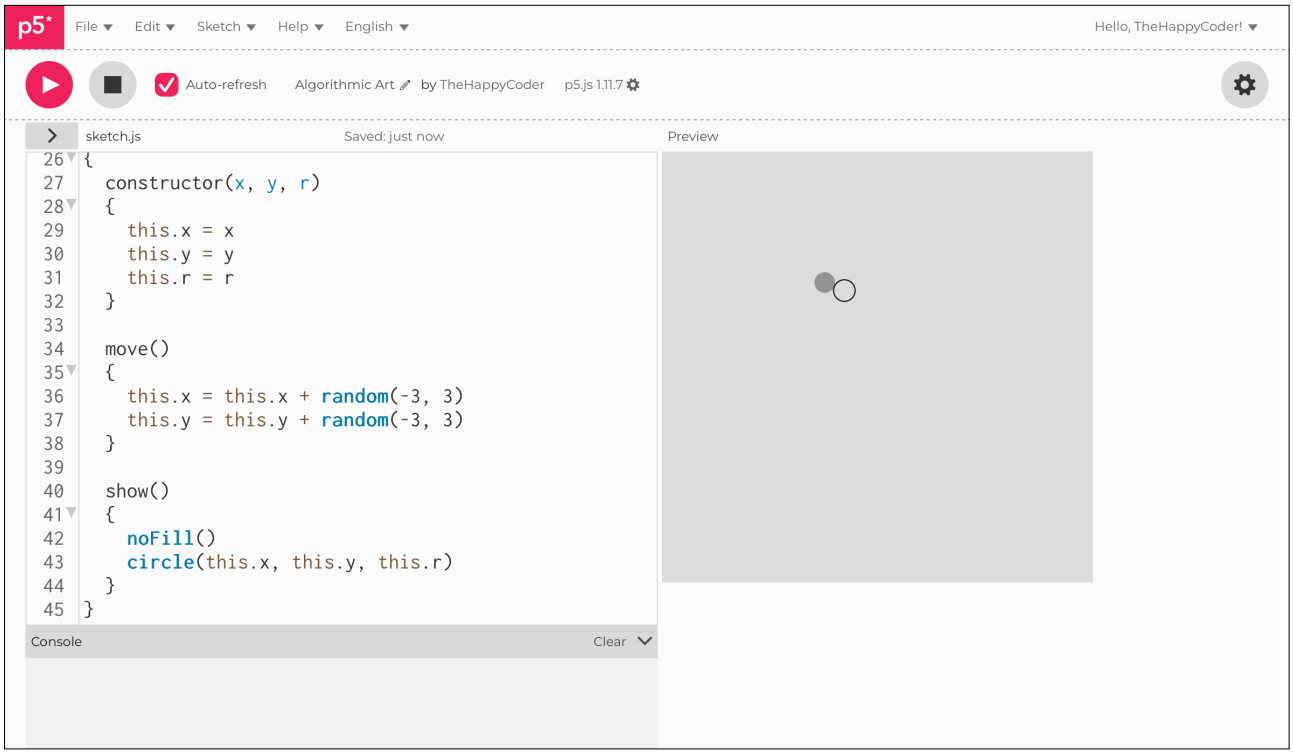
```
let bubbles = []
let bubble

function setup()
{
  createCanvas(400, 400)
}

function mousePressed()
{
  bubble = new Bubble(mouseX, mouseY, 20)
  bubbles[0] = bubble
}

function draw()
{
  background(220)
  for (let i = 0; i < bubbles.length; i++)
  {
    bubbles[i].show()
    bubbles[i].move()
  }
}

class Bubble
{
  constructor(x, y, r)
  {
    this.x = x
    this.y = y
    this.r = r
  }
```

```
  move()
  {
    this.x = this.x + random(-3, 3)
    this.y = this.y + random(-3, 3)
  }

  show()
  {
    noFill()
    circle(this.x, this.y, this.r)
  }
}
```

Notes

We create a new bubble every time we click on the canvas, but only one at a time. We want to keep each one, not discard the last one.

Figure D3.13



Sketch D3.14 when push comes to shove

We use a `push()` function to push each `bubble` into the array. Now we will keep all the `bubbles` we create with the mouse.

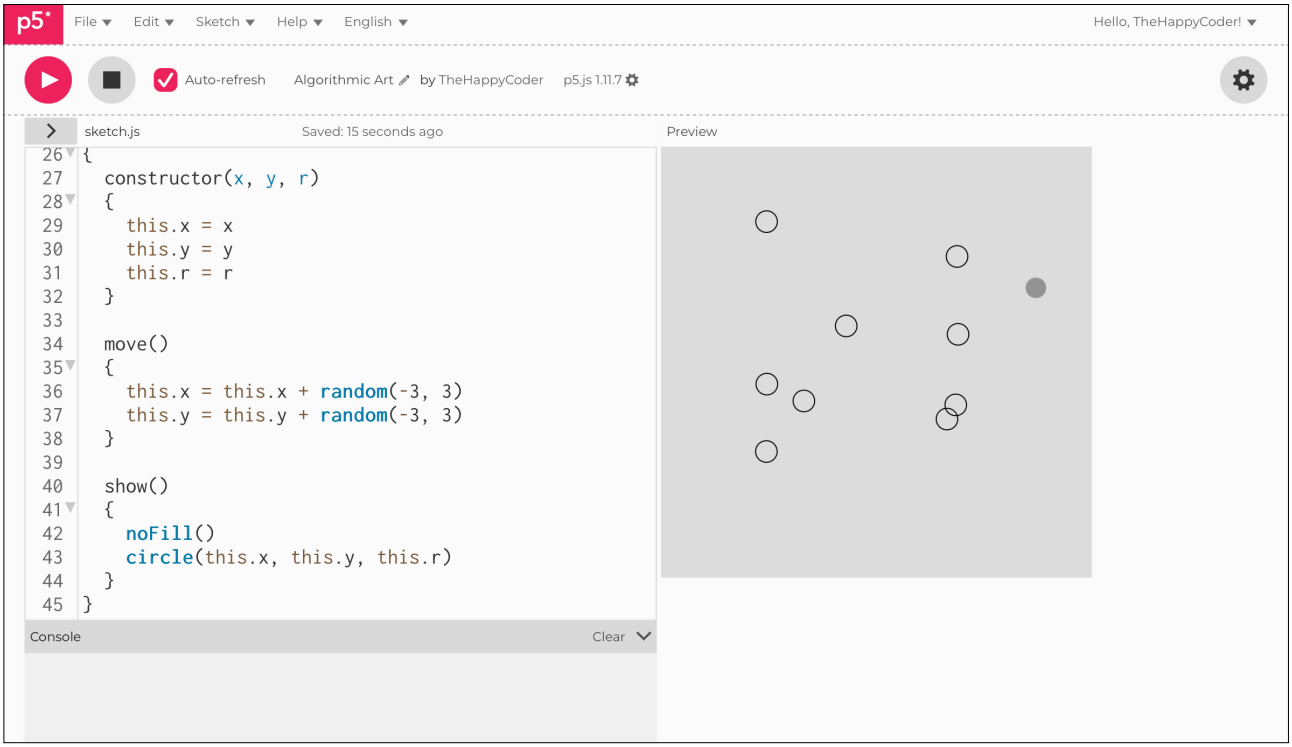
```
let bubbles = []
let bubble

function setup()
{
  createCanvas(400, 400)
}

function mousePressed()
{
  bubble = new Bubble(mouseX, mouseY, 20)
  bubbles.push(bubble)
}

function draw()
{
  background(220)
  for (let i = 0; i < bubbles.length; i++)
  {
    bubbles[i].show()
    bubbles[i].move()
  }
}

class Bubble
{
  constructor(x, y, r)
  {
    this.x = x
    this.y = y
    this.r = r
  }
```

```
  move()
  {
    this.x = this.x + random(-3, 3)
    this.y = this.y + random(-3, 3)
  }

  show()
  {
    noFill()
    circle(this.x, this.y, this.r)
  }
}
```

Notes

Now when you click on the canvas, each bubble remains on the canvas because they have been added to the array of bubbles.

Figure D3.14



Sketch D3.15 a bit of a drag

Finally, using `mouseDragged()` rather than just `mousePressed()`, we can make lots of `bubbles` as we click and drag the mouse.

```
let bubbles = []
let bubble

function setup()
{
  createCanvas(400, 400)
}

function mouseDragged()
{
  bubble = new Bubble(mouseX, mouseY, 20)
  bubbles.push(bubble)
}

function draw()
{
  background(220)
  for (let i = 0; i < bubbles.length; i++)
  {
    bubbles[i].show()
    bubbles[i].move()
  }
}

class Bubble
{
  constructor(x, y, r)
  {
    this.x = x
    this.y = y
    this.r = r
  }
```

```
  move()
  {
    this.x = this.x + random(-3, 3)
    this.y = this.y + random(-3, 3)
  }

  show()
  {
    noFill()
    circle(this.x, this.y, this.r)
  }
}
```

Challenges

1. What could you create with a few tweaks?
2. How would you create random shapes?

Figure D3.15